Secure Lightweight Entity Authentication with Strong PUFs: Mission Impossible?

Jeroen Delvaux[1,2], Dawu Gu[2], Dries Schellekens[1], Ingrid Verbauwhede[1]

1. University of Leuven (KU Leuven) and iMinds, Belgium
2. Shanghai Jiao Tong University, China

CHES 2014, Busan

Contents

- Mission: Secure Lightweight Entity Authentication

- Physically Unclonable Functions (PUFs). How can they help?

- 8 PUF Entity Authentication protocols
  - Security and practicality analysis
  - No protocol details here, only properties (*limited presentation time*)

- Conclusion

Lightweight Entity Authentication

2 parties sharing a secret

e.g. building access card

IC (CARD) *prover*

- Resource-constrained
- Low-cost

verifier SERVER

- Secure storage
- Computational power

protocol

Side-channels
Fault injection
Invasion

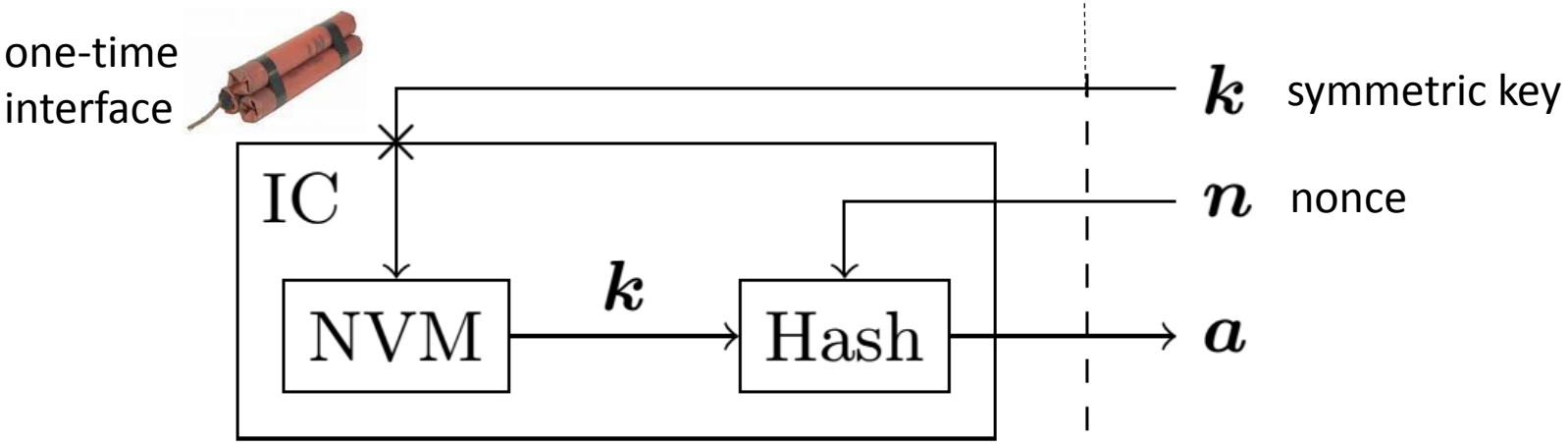
Eavesdropping
Manipulation
Replay

SECURITY = HARD

SECURITY = EASY

Lightweight Entity Authentication with PUFs?

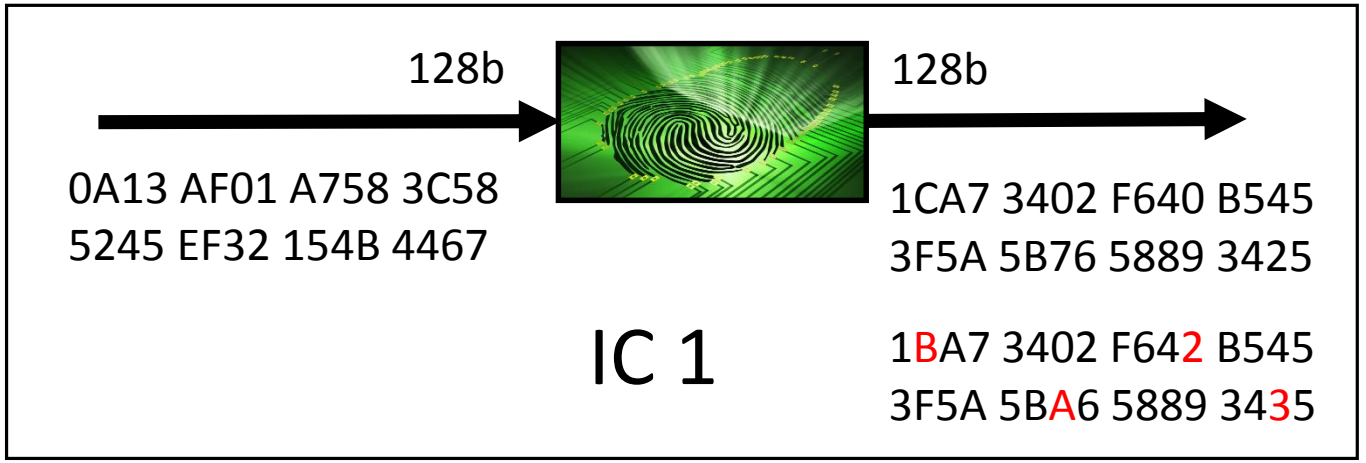
Conventional approach: Non-volatile Memory (e.g. Flash)



NVM:
High manufacturing cost
Flash floating gate ≠ CMOS compatible
PUFs = CMOS compatible

NVM:
Vulnerable to physical attacks
Robust electrical storage
PUFs = chemical storage

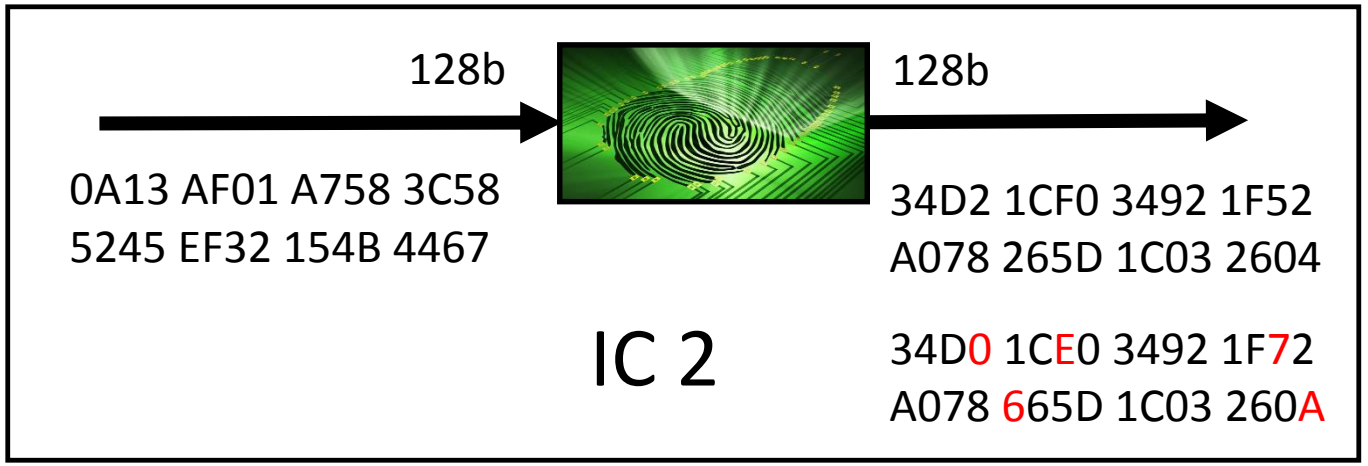
PUFs = Physically Unclonable Functions



Chip-dependent binary function with noisy output

Evaluation 1

Evaluation 2 ≈ 1-15% noise



Similar to biometrics, applied to an IC rather than a human

Evaluation 1

Evaluation 2 ≈ 1-15% noise

PUFs = Physically Unclonable Functions

Does not exist, but **all 8 protocols** need it to counteract brute-force and random guessing

<u>The protocols need:</u>

"Strong PUF"
(e.g. arbiter PUF)

+ lightweight solution for output expansion: repeated evaluation e.g. *Out* = PUF(PRNG(*In*))

Note: For convenience, we define strong PUFs using the popular more recent notion of large input space rather than the original definition

<u>The protocols cannot use:</u>

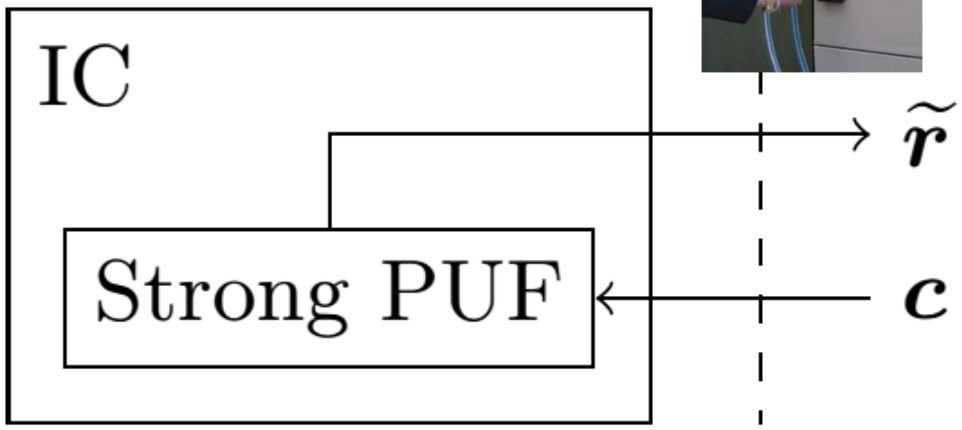
"Weak PUF"
(e.g. SRAM PUF)

(This type of PUF is mainly used to generate a secret key)

Our reference protocol

Entity Authentication: Basic strong PUF protocol

Simple challenge-response



54C…0E1 ≈ 55C…0E1

46B…3A4

Prerecorded I/O pairs	
1F2…51F	C73…467
46B…3A4	55C…0E1
E03…127	0A1…245

3 authentications, discard pair after use to avoid replay

Very efficient hardware but…

No secure instantiation: PUF modeling attacks

- PUF I/O behavior is correlated
- Mathematical clone: learn full I/O behavior given a small training set (machine learning)
- No PUF has valid claim to be resistant and lightweight

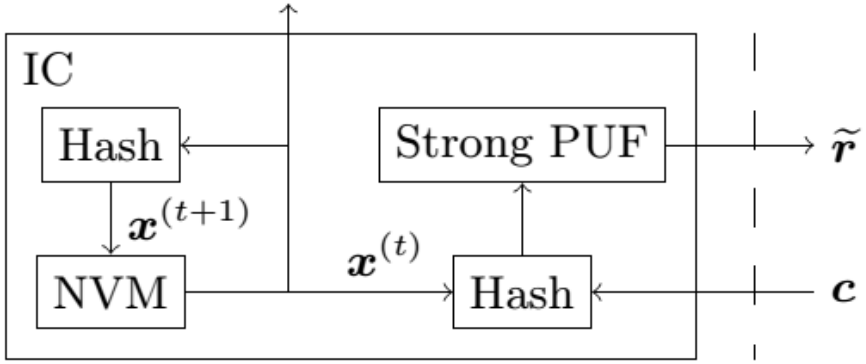
All 7 other protocols have additional building blocks: Hash, TRNG, Error-Correcting Code, …

8 Strong PUF Entity Authentication Protocols

- Basic (2001, PhD MIT)
- Controlled (2002, ACM CCS)
- Öztűrk et al. (2008, PerCom)
- *Hammouri et al.* *(2008,* journal*)*
- Logical Reconfiguration (2011, CHES & journal)
- Reverse Fuzzy Extractor (2012, FC)
- Slender (2012, SP & journal)
- Converse (2012, DATE & TRUST)

OUR FIRST CONTRIBUTION: FIRST OVERVIEW

Common framework with reference, same notation (IC block diagram & protocol), initiate comparison



(g) Logically reconfigurable PUF [11].

	Token		**Server**
(1×)	Write-secure NVM: $x^{(t)}$	⟵	$\langle c_i, r_i \rangle$, with $i \in [1\ d]$
		⟷	and $c_i \xleftarrow{U} TRNG()$
Auth. (d×)			$\langle c, r \rangle \leftarrow \langle c_d, r_d \rangle$
		$\xleftarrow{c}$	$d \leftarrow d - 1$
	$\widetilde{r} \leftarrow PUF(Hash(x^{(t)}, c))$	$\xrightarrow{\widetilde{r}}$	
			Abort if $HD(r, \widetilde{r}) > \epsilon$

8 Strong PUF Entity Authentication Protocols

OUR SECOND CONTRIBUTION: FIRST ANALYSIS

Security of the Protocol

Impersonation?

Denial-Of-Service?

The next 7 slides
Severe issues for all-but-one protocol.
We do not recommend their usage

Practicality of the Protocol

Are the PUF advantages (w.r.t. NVM) preserved?
- Low-cost manufacturing
- Improved physical security

PUF imperfections taken into account?
- Noisiness
- Prone to modeling

Are there PUF assumptions degrading the usability / generality ?

Efficiency & Scalability

PUF noisiness taken into account?

Error Tolerance

Basic 

Reconfiguration 

Hammouri et al. 

Slender 

Design flaw: error amplification. Protocol does not function. No FPGA proof-of-concept.

Error Correction

Our reference protocol 

Controlled 

Öztűrk et al. 

Reverse fuzzy extractor 

Converse 

Exhaustive search for error pattern by server. They assume noise < 1%. Might not be feasible for high noise. No FPGA Proof-of-concept.

Prevention of PUF modeling attacks?

Crypto (Hash)

Controlled

Reverse fuzzy extractor

Converse

Our reference protocol

Guarantee

Lightweight Protection

Öztűrk et al.

Hammouri et al.

Slender

PUF needs high resistance

No protection

Basic

Reconfiguration

Not usable, no secure instantiation

PUF advantages w.r.t. NVM preserved? Denial-Of-Service?

No NVM

Basic

Controlled

Reverse fuzzy extractor

Hammouri et al.

Slender

Converse

Our reference protocol

Write-Secure Reprogrammable NVM

Reconfiguration

Read/Write-Secure Reprogrammable NVM

Öztűrk et al.

Undermines the advantages of PUF technology: low-cost manufacturing & physical security.

NVM = state vector (requires synchronization between PUF-IC and server). No user authentication for state update. Attacker can do it too: DoS.

PUF output expansion exploits?

We did not spot a problem

Basic — Hammouri et al.

Controlled — Reconfiguration

Öztűrk et al. — Converse

Not applicable (weak PUF)

Our reference protocol

Token impersonation (for the chosen implementation only)

Reverse fuzzy extractor — **PRNG = LFSR**

Slender — **PRNG = LFSR XOR LFSR**

PUF assumptions degrading the generality?

Every strong/weak PUF

Our reference protocol

Every strong PUF

Controlled

Converse

Every strong PUF satisfying the assumptions

Basic	**Robust against modeling (does not exist)**
Reconfiguration	**Robust against modeling (does not exist)**
Öztűrk et al.	**Both easy & hard to model (contradiction)**
Hammouri et al.	**Both easy & hard to model (contradiction)**
Reverse fuzzy extractor	**Noise lower limit (opposing stability)**
Slender	**Both easy & hard to model (contradiction)**

Efficiency / Scalability (server storage)?

Constant w.r.t. # authentications

Our reference protocol

Öztűrk et al.

Hammouri et al.

Reverse fuzzy extractor

Slender

Linear w.r.t. # authentications

Basic

Controlled

Reconfiguration

Special category

Converse

Server and attacker face the same brute-force workload. Not usable.

Comparison

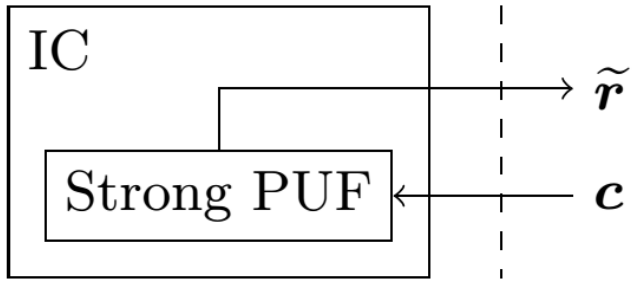
- Resources
- Authenticity Type
- Server storage

	Weak PUF	*Strong PUF*[1]	*NVM*	*TRNG*	*Gen*	*Rep*	*Hash*	*1× interface*	*Other*	*Token auth.*	*Server auth.*	*# Auth.*	*CRPs*	*Model*	*Key*
Reference I	×	×	✓	×	×	×	✓	✓	×	✓	×	∞	×	×	✓
Reference II	✓	×	×	×	×	✓	✓	✓	×	✓	×	∞	×	×	✓
Naive	×	✓	×	×	×	×	×	×	×	✓	×	d	✓	×	×
Controlled	×	✓	×	×	×	✓	✓	✓	×	✓	×	d	✓	×	×
Öztürk et al.	×	✓[2]	✓[6]	×	×	×	×	✓	✓	✓	×	∞	×	✓	×
Hammouri et al.	×	✓[2]	×	✓	×	×	×	✓	✓	✓	×	∞	×	✓	×
Reconfiguration	×	✓[3]	✓[6]	×	×	×	✓	×	✓	✓	×	d	✓	×	×
Reverse FE	×	✓[4]	×	×	✓	×	✓	✓	✓	✓	✓	∞	✓	×	×
Slender	×	✓[5]	×	✓	×	×	×	✓	✓	✓	×	∞	×	✓	×
Converse	×	✓	×	✓	×	✓	✓	✓	✓	×	✓	∞	✓	×	×

[1] Including response expansion.
[2] Easy-to-model.
[3] Robust against modeling.
[4] Non-determinism lower bound.
[5] Both easy- and hard-to-model.
[6] Reprogrammable.

Conclusion

- Secure Lightweight Entity Authentication with Strong PUFs: Mission Impossible?
- PUFs seem too brittle to be used without additional crypto (hash, …).
- PUF only one component in security architecture: still need TRNG, …
- Breakthrough: strong PUF with strong cryptographic properties (no machine learning)



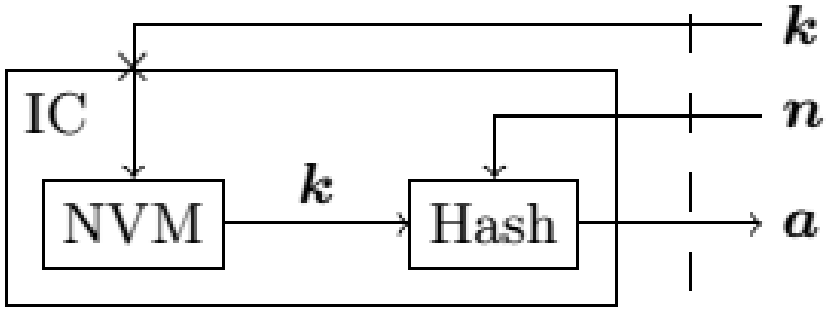
Not so very optimistic

- Decade of research: no success. History of Machine Learning Attacks
- Infinity of output bits with limited number of circuit elements
- Most promising ideas (e.g. optics) are not lightweight
- Unavoidable trade-offs: security vs noise

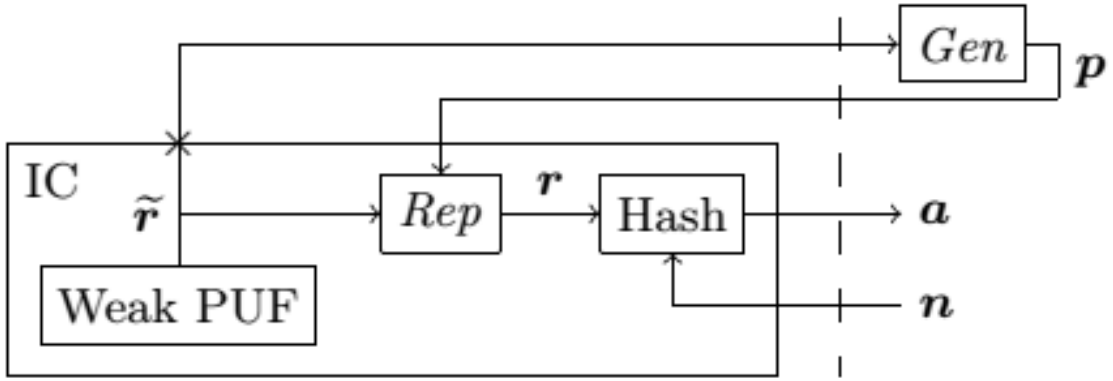
- Coming soon: extended version on IACR Eprint (including 3 more protocols)

Thank you! Questions?

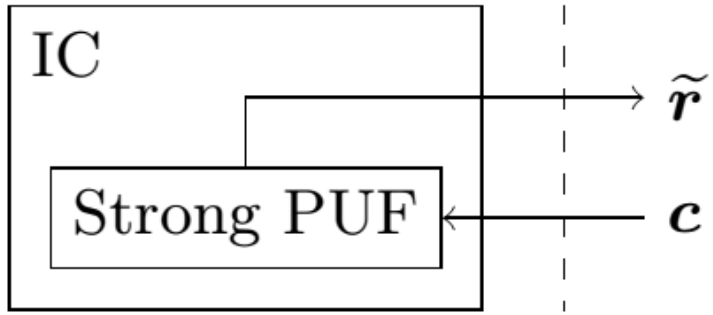
Appendix – Protocol Figures



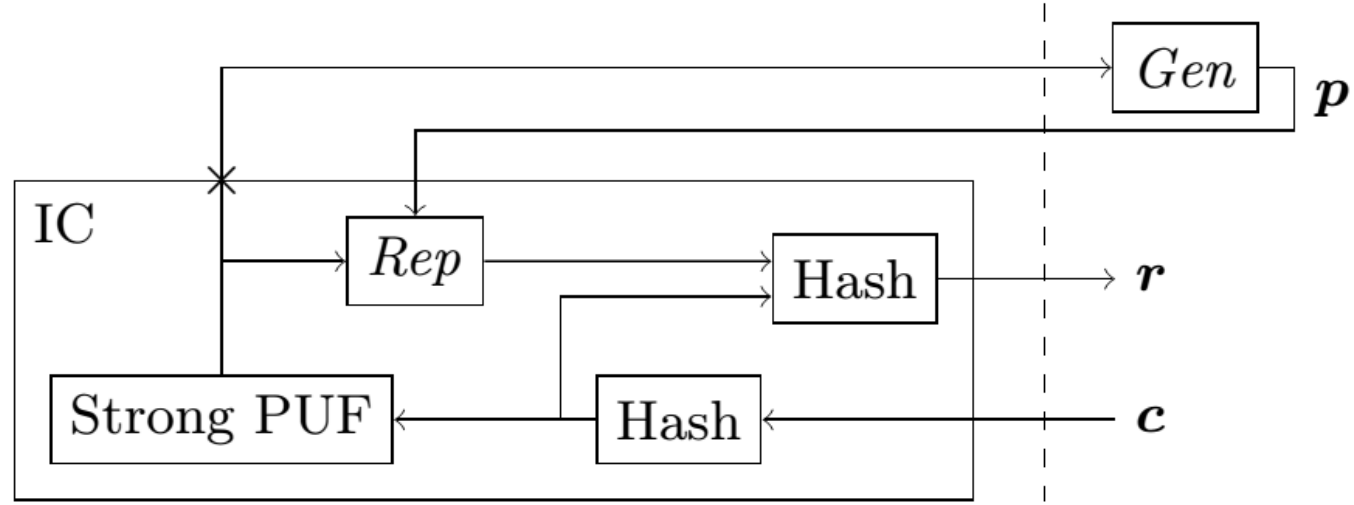
(a) Reference I: key in NVM.



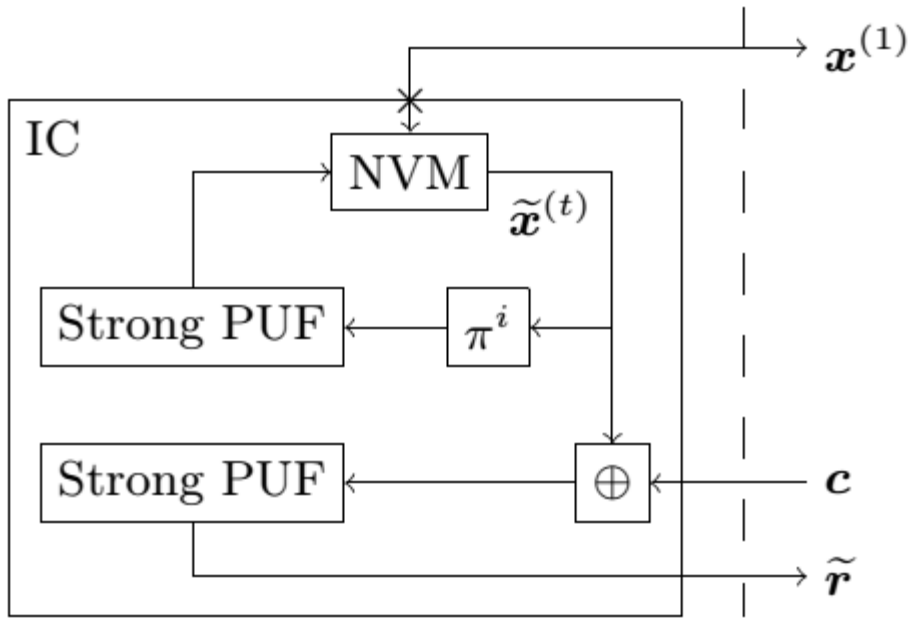
(b) Reference II: key via PUF and FE.



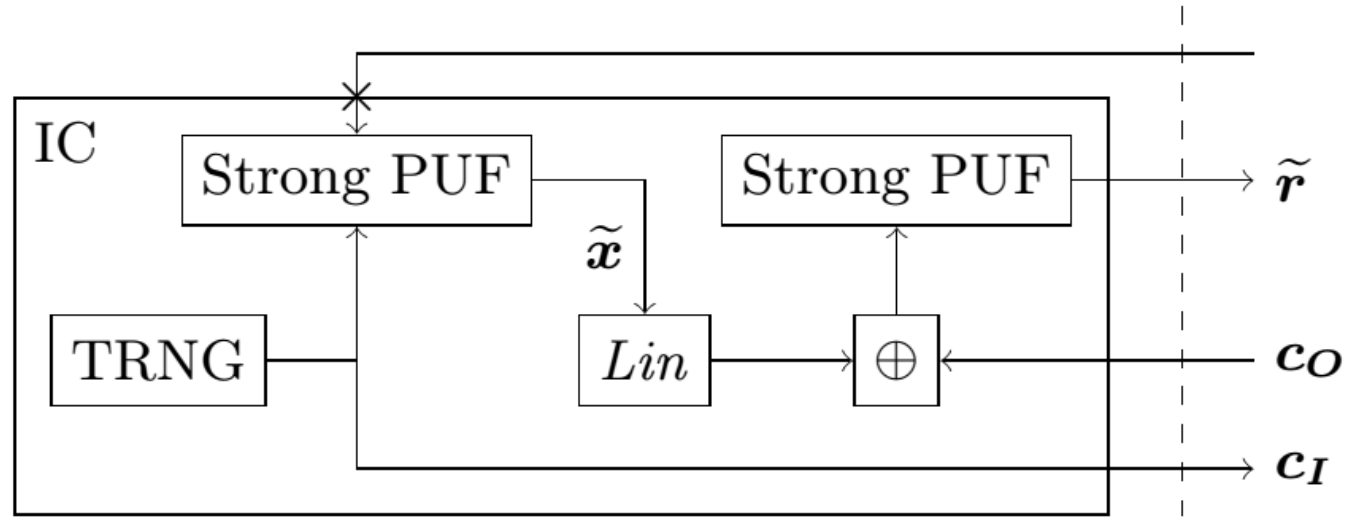
(c) Naive strong PUF [18].



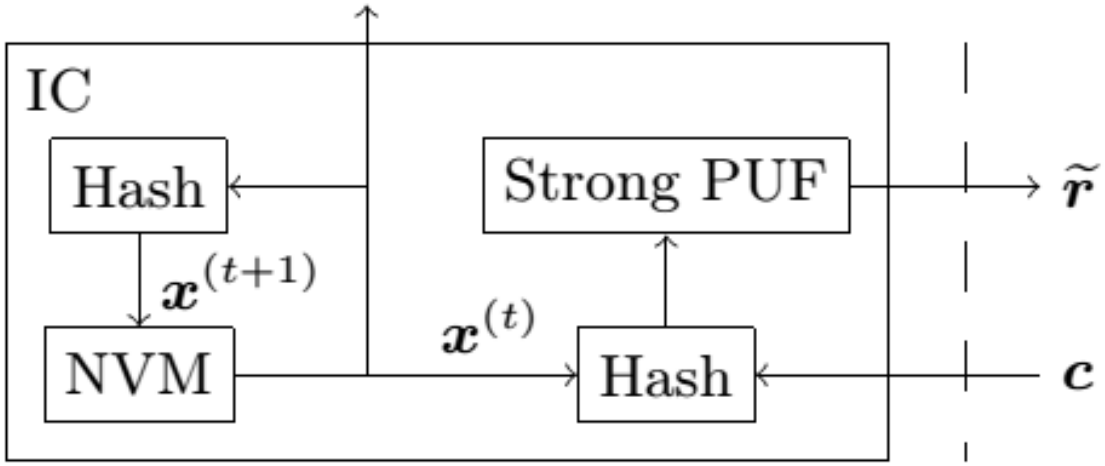
(d) Controlled PUF [4–6].



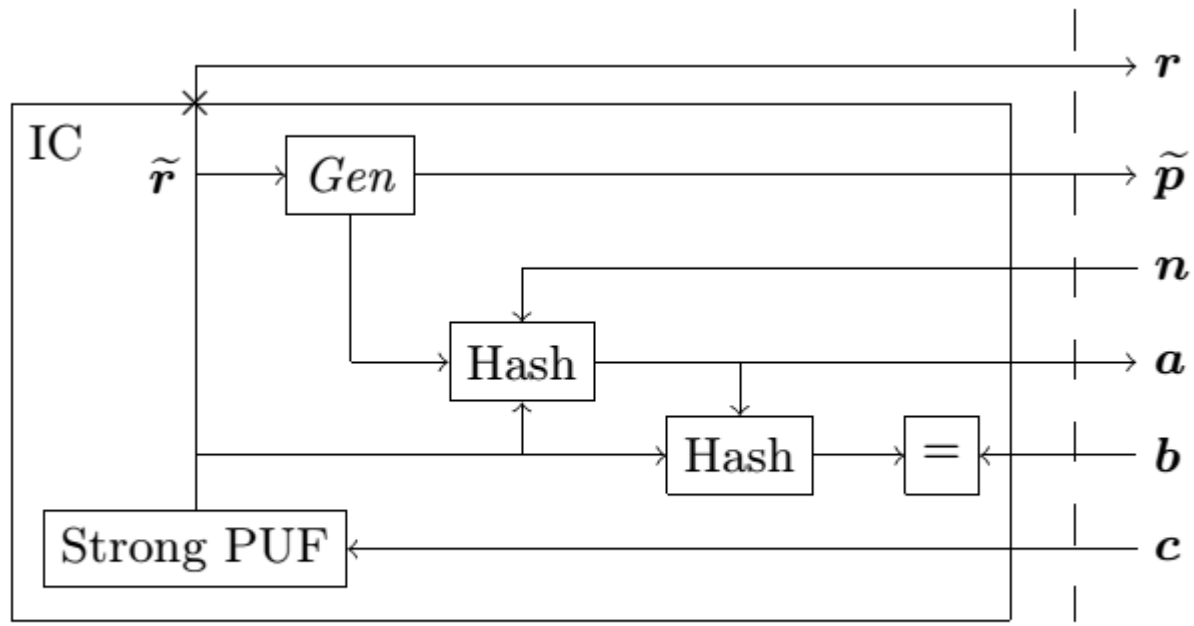
(e) Öztürk et al [17].



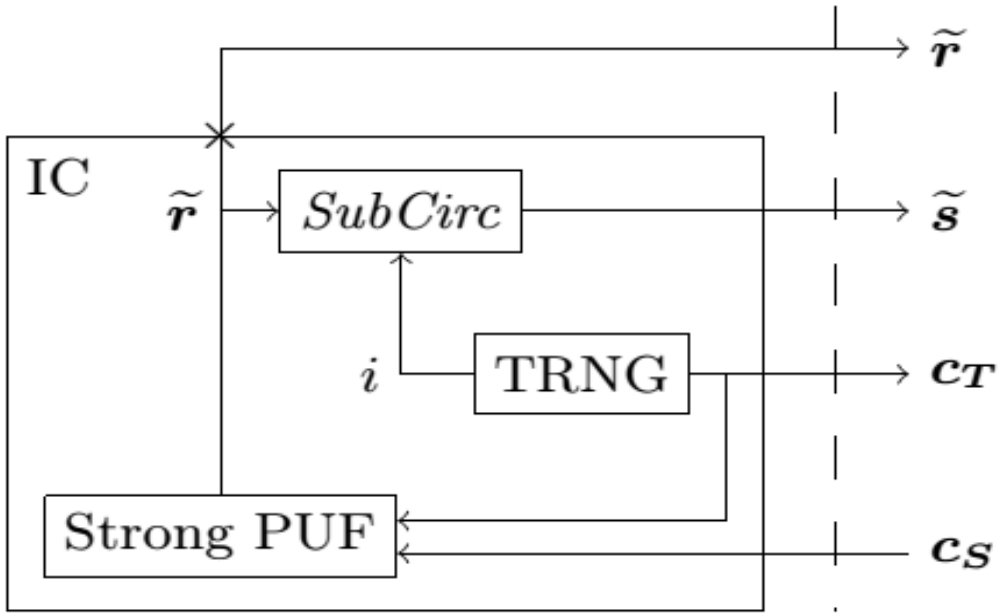
(f) Hammouri et al [8].



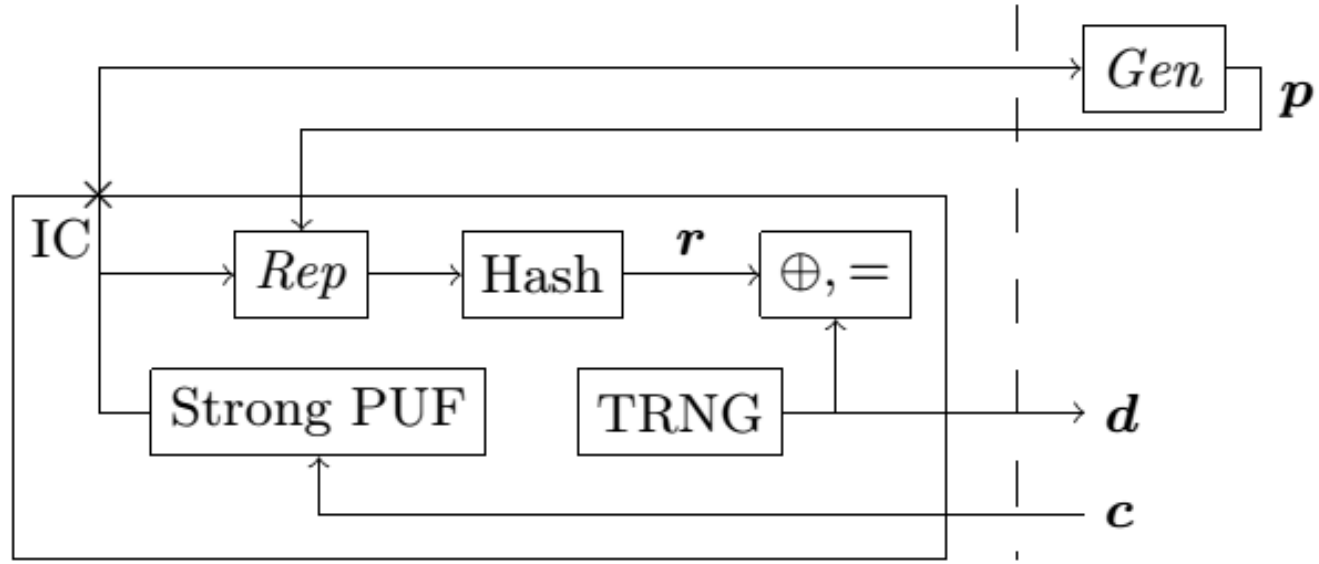
(g) Logically reconfigurable PUF [11].



(h) Reverse FE [24].



(i) Slender PUF [16].



(j) Converse [12].